DATA STREAM:
A SOFTWARE THRILLER

By Steve Jackson

Draft OCT 29, 2014

FOREWORD

In the heady days of WWII, the Lockheed Martin Aircraft Corporation in Burbank, California was commissioned by the US Air Force to develop a high speed, high altitude jet fighter to counter the increasing German jet aircraft threat. Its response was to set up the Advanced Development Projects Division. A select group of individuals was assembled to concentrate on development of what would become the P-80 Shooting Star, the first operational jet fighter used by the US Air Force. It was designed, built and delivered in just 143 days.

A tent set up next to a plastics factory became the incubator for the team and its special "project". Fumes that wafted in from the plastic factory made for an obnoxious atmosphere or skunk-like smell that nobody really appreciated working in. As a result it was nicknamed the Skunk Works, a name that has since become an official alias for the division. The Skunk Works has changed locations and gone on to develop a number of significant aircraft for Lockheed.

From its early beginnings, a "Skunk Works or Skunkworks" project has come to signify a project of radical innovation developed by a select team or group of people who do all the research and development. It is accomplished with a high degree of autonomy and often secrecy. It occurs outside of the mainstream of general R&D and manufacturing. Bureaucratic interference is minimized or even eliminated for this group and its project(s).

The original "Skonk Works" (as it was called) was a fictitious moonshine factory in the popular Li'l Abner comic strip.

TABLE OF CONTENTS

CHAPTER ONE

In Which

We Are Introduced to Norman Swift

and the Story Begins

It was a dark and stormy night. The clock struck one as Norman Swift was putting the final touches on his plan for his first Skunkworks project. He was excited – at long last, he was ready to test his cherished concept for a ***high-performance software development paradigm***. He rolled the word paradigm off his tongue a couple of times. Was he being audacious in thinking that his concept amounted to a new paradigm in software development and implementation? He had mulled it over and over; he was convinced it was.

Norm mentally reviewed the last ten software projects at Norman Swift & Associates. All had been disappointing, some disastrously so. In fairness, two were not awful, because they were cancelled before the awfulness became apparent! He wondered for the thousandth time why he was in the software game. "Because I need to be in some game", he muttered. "At least it's warm and dry here."

Norman had identified the root problem long ago – ***Requirements Drift!***

- We never get to design, build, test and deliver a project without ***late-arriving change orders*** tearing apart the fabric of our work.
- This happens because of the ***"I'll Know It When I See It"*** phenomenon. You deliver a feature as specified and the user realizes that it is not what they need. "Keep trying until I know it's right."
- This happens because business is complex and intricate, always in flux.

BUT, we know these things, and they are not going to change, so why do we not have a flexible development paradigm that ***welcomes*** late-arriving change orders?

Why are we always surprised and devastated by the inevitable?

Duh!

Norman sighed. That sometimes seemed to help.

CHAPTER TWO

In Which

Norman Launches the Skunkworks

The crew straggled in to the conference room just before 9 am. They were Eric, Brian, Ann, and Phil. They represented some 35 years of accumulated software development experience. Norm had the highest respect for their talents and the collective potential they brought to the table. If anybody could understand what he was about to bring forth and then run with it, they could.

Norm pushed the door closed and drew a deep breath. “OK, team; I have good news and bad news.”

Brian – “Bad first.”

Norm – “The bad news is that we lost the Ziffer bid, so I have no funded work for you guys.”

Eric – “That's pretty bad news, Norm. Not much room left for the good news.”

Norm – “The good news is that I have long wanted to do a Skunkworks project, and I am gonna launch it now during this slow period. I want you guys on my team. The funding is internal, since there is no client money for this.”

Eric – “What if we had asked for the good news first?”

(laughter)

Ann – "No offence Norm, but this is not a rich company. How long can the Skunk go on nothing but internal financing?"

Norm – "We have four weeks. A month from now we will have developed the reference application of a shiny new software development paradigm that I call ***Datastream*.**"

Phil – "And we can do all that in four weeks you say? Norm, are you new here?"

Norm – "Guys, this is a powerful concept that I've been mulling over for many years. Now is the time to launch it. It's now or never. In 5 weeks I will be a prophet, or else the newest greeter at a Wally's."

Eric – "OK, Boss, what is this new secret sauce?"

Norm – "It's real simple – just two rules:

One, we will record everything of interest, indelibly and unambiguously, in Datastream records. That will be the sole enduring storage method available.

Two, all the computing will be done in engines whose application is to read and write Datastream records.

That's the new paradigm in a nutshell."

Ann - "That's it? That's your inspiration? It's kinda thin. This sounds like the unit record punch cards in the 1950's. But we've come a long way since then, Norm."

Norm - "Well, if this paradigm is as good as I claim, it will be apparent from week one. You won't have long to wait."

Phil – "What's the reference application? Something simple, I hope."

Norm - “Not so simple. I call it **Acme Rentals** – an equipment rental chain of 100 outlets. They trade tools and supplies among stores and they need to track rental and return times. Their asset base is over 100,000 items, big and small. It's a significant logistics challenge, for sure! PLUS they need business intelligence.”

Eric - “OK, but this is a job for the Enterprise Resource Planning giants, not for us. Jobs like this consume a million man-hours.”

Norm - “Yes, I agree with you Eric. The normal paradigms do need that kind of effort. But this is a new broom. A Skunkworks. If the reference application were smaller, the victory would be less dramatic.”

Ann - “So this is a David vs. Goliath kind of deal?”

Norm - “Yup.”

CHAPTER THREE

In Which

The Datastream Records Are Specified

Norm - "OK, let's get started. Here's what we're gonna do. Make up a project spreadsheet with a different sheet for each type of Datastream record. These Datastream records will be indelible, meaning that they are read-only after creation.

So how are errors corrected? By means of a new Datastream record to be published when the error is detected. Them's the rules.

We need to decide what is **of interest** to Acme. First and foremost, there is the **Asset Record**. It describes each asset item in unambiguous detail. So there needs to be the Serial Number, Manufacturer, Date of Acquisition, Description, Price paid, etc."

Eric - "Not certain that I agree with you there, chief. Your date of acquisition implies this item is unique, whereas Acme has several of everything. Best to have an Asset Descriptor Record that is prototypical of the asset model. So it just needs Acme Inventory Number, Model Number, and Manufacturer. Other traits like Serial Number, Colour, Date of Purchase, Price Paid, etc. can be in a different Record – call it **Item record**."

Norm – "Good observation, Eric. Well, the record names can change as we mature here and the big picture becomes clearer. What's important is that the sheets each include the fields that are necessary for that record to be authoritative and unambiguous."

Norm - "Ann will maintain a central Dictionary of the Datastream records. We have to

protect the integrity of the field definitions closely."

They made steady progress. Each Datastream record formation issue was decided by Acme's business policies and rules. It took most of the day to codify all (or most) of these rules.

At 4:30 pm Norm summarized the day.

Norm - "OK Team, we are 5 person-days into the project and already we have laid the keel for the good ship **Acme Rentals,** because we have defined some 27 key Datastream record types."

1. ITEM DESCRIPTION
2. ITEM PURCHASE
3. ITEM RENTAL
4. ITEM SALE
5. ITEM REPAIR
6. ITEM LOSS
7. CLIENT PURCHASE ORDER
8. STAFF PAYROLL ITEM
9. STAFF VACATION ITEM
10. EXPENSE PAYOUT
11. WHIMIS REQUEST
12. ITEM USER MANUAL REQUEST
13. CAPITAL EXPENSE – REPAIRS
14. ITEM INTER-STORE TRANSFER
15. RENTAL OPPORTUNITY LOST

16. ITEM RESERVATION REQUEST
17. ITEM RESERVATION GRANT/DENY
18. UTILITY METER READING
19. DOOR ACTIVITY
20. SECURITY ALARM ACTIVITY
21. STAFF TIMECLOCK ITEM
22. VEHICLE ITEM
23. FUEL PURCHASE
24. LUBRICANT PURCHASE
25. SUPPLIES PURCHASE
26. VENDOR DEMOGRAPHICS
27. CLIENT DEMOGRAPHICS

Phil - "I like this process, Norm. It seems like we are actually accomplishing something here."

Brian - "Yeah, because this foundation work will endure regardless of specification change orders down the road."

Ann - "Not to be negative, Norm, but all we've done is list these records as being 'of interest' to Acme Rentals. We have not done the work to create even one of these."

Phil - "Ann, you're looking at this wrong. The Datastream paradigm is supremely liberating because it separates the creation from the storage from the use of these records. We can proceed with these tasks in any order, and with good confidence that we are contributing to the final solution. We've never had that before!"

Ann - “Hmmm.”

Eric – “Norm, this is the first day at Swift & Associates that I've truly enjoyed. We have made solid progress with good harmony. Every disagreement has been resolved using this method and nobody got angry.”

They broke at 5 pm.

CHAPTER FOUR

In Which

The Datastream Schema Is Designed

On Tuesday morning Norman launched his team into Day 2.

Norm - “It's time to build the Datastream infrastructure so we can make the data we have defined available to the engines we are going to build. I want this to be a cloud application and we are going to use a standard database product to store the data.”

Ann - “I nominate MYSQL. It's free, sophisticated and there's lots of help out there.”

Norm - “I agree Anne. MYSQL it is.”

Brian - “As I see it we first need tables to store the data about the data. That is the metadata. I see several tables. The first describes the data record - an identifier, a record descriptor and a tenantID so we can accommodate more than one company. The second describes the fields in the record. This will include: a record type (which is the link to the identifier in the record descriptor), the field number, field name, length, type, a description and the requisite tenantID.”

Ann - “That seems simple enough. What about storing the data?”

Brian - “I see two tables: The first stores the record information (a unique key, a record type that links to the identifier in the record descriptor, a date time to tell when the record was created, a status to indicate if this is an original record or a modification of a previously created record, the creator's identifier and the tenantID. The second will hold the actual data and will have a record for each field. It will have a unique key, a record number that links back to the record

information table's key, the field number, the data and the tenantID."

Norm - "Let's store the data in strings . A string can characterize any data and I'm not worried about storage efficiency here. You can buy a terabyte disk for $100."

Eric - "There's going to be a lot of data here Norm. Storage isn't a problem as you say, but search times might be."

Brian - "Let's generate working tables for each data type. The user interface can work with the working tables and the Datastream will mirror what goes on in these tables. The tables will contain only the data of a specific type so we will divide and conquer."

Eric – "If we keep the Datastream and the working tables synchronized so that when we change one, we change the other, we can reduce the times we have to re-construct the working table. This will make things peppier."

CHAPTER FIVE

In Which

The User Interface is Started

Later Tuesday morning Norman got into the GUI.

Norm - “So how are our clients going to interface with our new paradigm?”

Ann - “Cloud. It has to be a Cloud application.”

Norm - “Why is that?”

Ann - “Well, we don't know what infrastructure our future clients will have. It could be Windows or an Apple OS or Linux. We don't know what the state of their hardware will be. All a cloud app needs is a browser and that's available on pretty much any system. Also, deployment will be butt simple (excuse my French). Web applications will scale with the enterprise and we can find multi-lingual support. The Cloud is where it's at these days.”

Eric - “What about security?”

Ann - “Cloud security is improving all the time. Access can be through an encrypted connection and the sign-on security will be the same as for an in-house application. The data will be housed in a data fortress and backed up by professionals and all this goodness happens without the client requiring an I.T. Staff.”

Norm - “O.K. You've convinced me. What web application will we use?”

Ann - “There are many. I would look for one that provides the easiest development process but has enough features to allow us to do some sophisticated things.”

Norm - “O.K. Anne. Do some research and let me know what you recommend in a few hours.”

. . . . . Later . . . . . .

Ann - “Norm, I think WaveMaker would be a good choice for us. It uses JAVA and JavaScript coding and has tools for deployment and database activities. The drag and drop table tools will allow us to quickly provide a user interface to the working tables and we can express the business logic in JAVA on the server.”

Norm - “I trust your opinion Anne. Let’s go with Wavemaker.”

. . . . . Even Later . . . . . .

Ann - “I installed WaveMaker and MYSQL and put up the preliminary database. I created an input screen for the Datastream metadata and I've loaded the data definitions that we decided on. I am also developing an application that will generate working tables from the data definitions.”

Norm - “Good work Anne. You're a dynamo!”

CHAPTER SIX

In Which

The First Engine Is Designed

On Wednesday morning Norman launched his team into Day 3.

Norm - "Yesterday we laid the keel of the good ship **Acme.** Today we will begin crafting and welding together the parts of the ship - the decks, pumps, engines, propellers, etc.

In our Datastream paradigm we will need logic routines to act on the Datastream records, in order to express Acme's business rules while conducting Acme's minute-by-minute business. I call these 'logic engines'."

Ann – "Norm you are waxing poetic about this design – likening it to a ship."

Norm – "Well, the software designs built using the Datastream paradigm will always be elegant. Even after the late-arriving change orders."

Eric - "OK, great. But what about the other types of business rules that cannot be expressed through these engines. How will that logic be expressed?"

Norm - "Don't worry about that happening. The set {of business rules that cannot be expressed through these engines} is a null set."

Norm - "The engines all use the same interface or API. Who can guess what that API is?"

Eric - "Haven't got the foggiest clue, Boss."

Ann - “Me neither.”

Norm - “Would you be surprised if I said that the API is the Read/Write of Datastream records?”

Eric - “You mean these engines can read and write Datastream records and that's all they can do?”

Norm - “Yes, exactly.”

Ann - “But where's the GUI located?”

Norm - “The user-facing GUIs are all inside engines.”

You could hear a pin drop.

Eric - “OK, and where's the master database? We have not even begun its design. So how can this work?”

Norm - “Sure there's a master database. That's what we did yesterday.”

You could hear a pin drop.

Eric began deep breathing, loudly. “*I'm having a panic attack over here!”*

Norm - “This is a new paradigm. This is business as UN-usual. Our “database” is the accumulation of the Datastream records. That's how we do things here. Trust me, Acme Rentals will be well served.”

Ann - “OK, how will these engines communicate with each other and with Datastream records? We're gonna need a message protocol.”

Norm - “The protocol is *Publish & Subscribe*. Each Engine **Subscribes** to Datastream records of specified Types, meaning the Records which carry data that are relevant to the engine's mission. Recall that each Datastream type is on a separate sheet.”

Phil - “And each engine writes Datastream records that are the result of its mission, to express the business rules of Acme Rentals?”

Norm - “Yes, exactly.”

You could hear a pin drop.

Ann - “But, nobody does computing like that these days. This is antiquated unit record thinking. It's 1950's stuff. We'll get fired!”

Norm - “This new paradigm is a resurrection of an old paradigm, empowered by modern technology.”

Eric - “OK, so give us an example of an engine that only does R/W of Datastream records.”

Norm - “Well the engines are inspired by the transactions that Acme performs during its daily business. Say I come to the counter and want to rent a claw hammer, which is Item #129 in the catalogue. We create and publish a Datastream record saying “Need 1 of #129 at Store 12 for 14:00 on Oct 9-12”. Because those are the facts of interest just now. No need to identify me as the client yet.”

Phil – “So we published a Datastream record. Then what do we do?”

Norm – “We wait.”

Ann – “And what are we waiting for, Norm?”

Eric – “I know! We are waiting for a Datastream record sent by an engine that subscribes to 'need' records, and will inform us as to availability of Item #129. It checks its hidden database of inventory and sends this Datastream record: ‘Send 1 of #129 SN 762345 to Store 12 for Oct 9-12’.”

Norm - “Correct! The whole population of engines can see the Datastream records, or the ones they subscribe to.”

Ann – “Where exactly are the Indelible Storage units located?”

Norm – “They can be located anywhere on the Internet, and the engines can be located anywhere, as well.”

CHAPTER SEVEN

In Which

The First Database Is Designed

Norm - "OK, so Phil I need you to make a database of the Items available for sale."

Phil - "But Boss, you said there would be no databases. That sure was a short-lived rule!"

Norm – "I said the Datastream records are the enduring, eternal store in our universe, and NOT the central relational database that everyone else uses. But we are free to make HIDDEN databases to facilitate the work inside our engines. 'Hidden database' means that the database is never exposed outside its host engine, and if it crashes it is easy to rebuild from the historical Datastream records."

Phil – "OK, I see the difference now. Still, we need a database analyst don't we?"

Norm - "No we do not need a database analyst. Because we are all capable of making small databases and that's all we will need. If we had a database analyst, she would take over the project and I can't let that happen."

Phil - "OK, I get it."

Norm - "Phil have you heard of 'Green Databases'?"

Phil – “Yes – that's when a database has just one user. It is very rare in practice.”

Norm - “Correct! And our hidden databases are Green. We do NOT share them with other users because there's no need. Each user will build his own hidden, Green database from Datastream records.”

Phil – “OK. But how am I supposed to make the database?”

Norm – “The Datastream records have recorded every event in Acme's business history. All the things that Acme owns were purchased, right? So you can build the database from the purchase database records. And you can assume that there's a start-up spreadsheet given to us.”

CHAPTER EIGHT

In Which

ACME's Business Intelligence Is Begun

Norm – "Brian, I want you to get started on the business intelligence for Acme."

Brian – "Now Norm, you have gone over the brink! How on earth can I do business intelligence before any database exists? Anything I write would be a guess and will certainly be scrapped later in the project."

Norm - "There you go again, assuming a central database when you know it won't exist. Central database is for THEM, not for US."

Brian – "I'm beginning to get your message. You're gonna tell me that the business intelligence will always come from the accumulated transaction Datastream records."

Norm – "Yup. The Datastream records will certainly capture all the transactions that business intelligence seeks to analyze, for the simple reason that the Datastream records CAUSED every transaction to happen. Nothing moves at **Acme Rentals** unless Datastream records cause the movement. So no detail can hide from Datastream analysis."

Brian – "Yes, that is a proof, Norm. OK, I will get started on business intelligence as you asked. And I will do the work with full confidence that I am contributing to the final solution, even though that solution has barely been sketched out!"

Norm - “That's the stuff, Brian!”

CHAPTER NINE

In Which

Acme's POS Is Begun

Norm - “Now, what shall we do for a Point of Sale (POS) system? Who has some ideas?”

Brian – “OPEN/POS does a nice job.”

Eric – “I think we need to do our own POS, in Datastream.”

Brian – “That's goofy. It takes years to do a POS system. I know, I've been there.”

Eric – “Granted. But now we're a Skunkworks, and we perform Miracles every day. Besides, what is a POS but what we've been doing from the start? All we need is a method to sum up the prices and collect payment.”

Brian – “I can't really argue with that, Eric.”

CHAPTER TEN

In Which

Acme's Loyalty Reward Program Is Begun

Norm - "Now we need a loyalty reward system. It pays $20 off, say, on every 5th rental invoice. Of course, it ties into the business intelligence system because it is an incentive for the clients to give us their ID. We can do a lot more with business intelligence if we know who our clients are."

Ann - "These transactions will have to be at the same retail store, right? We can't give credit for transactions system-wide."

Brian - "Now Ann, why would you say that?"

Ann - "Because it's obviously true! If we want system-wide credit then we will have to launch a loyalty server, or sign up with a commercial service."

Eric - "How about this concept – let's build a loyalty engine and have it subscribe to all the sales data. It will certainly be system-wide, because Datastream is system-wide!"

Ann - "OK, I see what you are saying. Yes, we can do that! The loyalty engine will publish a reward Datastream record when it calculates that a $20 reward has been earned, because this is the 5th transaction for this client. And the POS system will subscribe to these reward Datastream records, and will deduct the $20 discount from the amount owing, if the Datastream record specifies this client as the recipient. Sure, that will be easy to build."

Norm - “Now Ann, why did you expect that Loyalty would be difficult?”

Ann - “Norm, just shut up.”

CHAPTER ELEVEN

In Which

Acme's WHMIS System Is Begun

Norm – “Now we need a Workplace Hazardous Materials Information System. Acme Rentals is required to inform clients about potential hazards with their rental equipment and supplies.”

Phil - “OK, this should be easy. All we need is an Engine to subscribe to the items rented and check a hidden database against WHMIS requirements. When we get a match, we publish a Datastream record with the pdf file name of the sheets that need to be printed and handed to the customer. These files will be on a server, I suppose.”

Norm - “OK Phil, get started with that approach and we'll see how it performs.”

CHAPTER TWELVE

In Which

Acme's Gift Card Program Is Begun

Norm - “Now we need a gift card interface. Who knows about that?”

Ann - “I worked on an interface last year. There are stored value cards and central lookup cards. Some systems require a high-security terminal at POS so the transaction messages are secured.”

Norm - “Sounds like the gift card interface can be a function of the POS engine, then.”

Ann - “Yes, we could do that. Or, we could make a gift card engine that happened to be running on the same PC as the POS Engine.”

Norm - “OK, so the Datastream records would be published the same, regardless of the method used?”

Ann - “Maybe. I'll get back to you on that.”

CHAPTER THIRTEEN

In Which

Acme's Payment System Is Begun

Norm – "Now we need a payment capability. That means cash, credit card, and open account terms for the trades. I suppose the POS Engine will either:

a) Publish a Datastream record to say that money was received in cash, or

b) Publish a Datastream record to say that money was received from a payment terminal at the retail counter and describe the tender type, or

c) Publish a record indicating the sale was compiled along with the authorizer's name.

d) Publish a request for open account credit from the A/R engine."

Ann - "In case (c), an A/R engine will subscribe to these requests and it will check against the account status, and the engine will create a Datastream record indicating that credit was extended or denied."

Eric - "How will we balance the cash?"

Phil - "Of course, the sum of the Datastream records with cash payment must be the net amount in the drawer."

Norm - "The cash float amount? It can be declared in a Datastream record at the start

of the day."

Brian - "Pickups and loans can be Datastream records through the day. And also payouts."

Norm - "So we are creating our custom POS system for Acme Rentals, bit by bit."

CHAPTER FOURTEEN

In Which

The Skunkworks Goes Home-Office

Norm – "OK now comes a big change. We're all gonna work from home . That will save the cost and time of commuting. We will meet Friday mornings at some central place. We can be in frequent contact by video Skype. We all have cable Internet at home.

Here are the productivity standards that I expect each member to meet. I want three new engines per week – even if they are prototypes that don't do much yet. But they are publishing and subscribing the specified Datastream records. Also, I want constant deliverables after week 2. That means a working system that is always running, and gains functions every week. Our deadline is 2 months. So we will have gained a factor of 25 in productivity!

If there is a big disagreement in methods then we will build both. No need for conflict in the Datastream universe. Because the cost is low, we can afford to be playful with engine logic."

Phil - "Woo Hoo!"

They all went home.

CHAPTER FIFTEEN

In Which

The Skunkworks Gains Additional Developers

By video Skype:

Norm – "We are falling behind with the engine production. How can we speed this up? Without killing ourselves, I mean. We support the '9-to-5 company goals'."

Eric – "How about Elance. We know how to specify the logic for a new engine, and we can declare the Datastream records as C strings. It is easy to test engines to see if they perform as required. So let's post the engine production on Elance."

Ann – "You know, that could work! If we identified 4 contract programmers and they could each build 2 engines per week, we'd soon have the 20 engines we need."

Norm - "Yes, at that rate it'd take 20/8 = under 3 weeks. Let's give it a try!"

CHAPTER SIXTEEN

In Which

New Fields Are Added To a DS Record

Ann - "I hate to be the bearer of bad news. But the record type 12 needs two new fields. Sorry to be disruptive."

Eric - "Not a problem Anne, just define a new Datastream record, type 122 and make it the same as Type 12, but append the 2 fields. Then the old engines will be compatible."

Phil - "This is great! This avoids the pollution of cell definitions that happens with normal record data bases. You know – the disaster when a field definition is altered but nobody knows exactly when or what is was before the change. That erodes confidence in historical data, and makes the database suspect, unreliable."

Brian - "Should the engine that publishes Datastream record type 122 also publish type 12, to be polite to the old subscribers?"

Eric - "Yes. Or, perhaps another engine could take on that task - i.e., to publish a type 12 whenever it sees a type 122."

CHAPTER SEVENTEEN

In Which

A New Engine Is Tested With Old Data

Norm - “We have refined several engines to work better, based on experience in the stores. How can we test these new engines?”

Ann - “Not a problem, Boss. We can simply feed them the historical Datastream records and watch what they do. In effect, we can re-live the past and see how these engines would have performed had they been in existence back then.”

Phil - “That is so cool! I don't think any other paradigm supports that capability.”

CHAPTER EIGHTEEN

In Which

A New Business Intelligence Strategy Is Mandated

Norm - “There's a new VP Sales at Acme and he wants a new business intelligence strategy. It's something weird he learned at B-School.”

Ann - “Not a problem. Whatever it is we can code a business intelligence engine for him.”

Brian - “But won't that upset the users of the existing business intelligence engine, to make this change?”

Norm - “Well there can be a dozen business intelligence engines. They do not collide. And they are all cheap to run. So everybody wins!”

CHAPTER NINETEEN

In Which

A Supervisor Engine Is Designed

Norm - “We have invented several dependencies, where an engine is waiting for a response from another engine. But what if that response does not come? We'll have a stalled business process and that is ugly!”

Ann - “Yes, I've been thinking about that danger. How about a supervisory engine that looks for over-due responses? Of course everything is visible in the Datastream stream.”

Norm - “OK. And we could also publish an ‘overdue Datastream record’ when an engine feels it has not received service in a reasonable time. The supervisor can then launch a new engine to do the required work.”

Ann - “And the helpdesk can subscribe to these alerts so they can make changes if needed, even before anyone knows there's a problem.”

CHAPTER TWENTY

In Which

A Publish and Subscribe Service Is Harnessed

Phil - "Norm, let me do the Publish & Subscribe infrastructure. I learned a lot about it at a previous job."

Norm - "OK Phil, go ahead. You will probably use an Open Source solution?"

Phil - "Yes, of course. Stay tuned."

CHAPTER TWENTY-ONE

In Which

Syslog Messages Are Launched

Brian - “We got a crazy request from a client. He wants to interface with Kiwi Syslog to send alerts when certain conditions are triggered. He knows the Datastream records he wants to use, and the trigger algorithms. I guess we can make a Syslog engine for him, eh?”

Ann - “You bet we can, Brian!”

CHAPTER TWENTY-TWO

In Which

Building Security Is Designed

Phil - "We got a request for building security. This means door access RFID cards, door switch monitoring, etc. Can we do this in Datastream records, or would this be polluting a financial system with foreign data?"

Norm - "No fear of pollution! All types of Datastream records are welcome. We have a huge capacity and the records are inherently small."

Ann - "In fact, building access can be a huge clue in an audit process. If stock goes missing, then it is helpful to know who came and went, and at what times and dates."

CHAPTER TWENTY-THREE

In Which

Disaster Recovery is Designed

Ann - "Every company needs a disaster recovery plan. With indelible Datastream records, it is always easy to re-establish all the hidden databases from the Datastream record history. So the company is up and running in no time! We can easily run disaster recovery simulations to test and demonstrate this capability."

CHAPTER TWENTY-FOUR

In Which

e-Discovery is Designed

Norm - “In the modern world any company can face the need for e-discovery. This can come from a lawsuit or a search warrant, say. This is similar to assured chain-of-custody. With the ‘read only’ capability of every Datastream module, it’s all there for review right back to square one.”

CHAPTER TWENTY-FIVE

In Which

Assured Chain-Of-Custody is Designed

Norm - “A class of clients will need assured chain-of-custody for legal reasons. We can achieve this by storing the published Datastream records in multiple synchronized repositories around the Internet. Some can be Third Parties. We can thereby assure that nobody ever had access to all the stored Datastream records and could have fiddled with them. In every case the Datastream records are indelible. They are write once, read many. The Lawyers will love that!”

CHAPTER TWENTY-SIX

In Which

ACME RENTALS Gains an Online Reservation System

Norm - “How should we approach online reservations?”

Phil - “We need a rental engine of course. It will present a browser interface to allow customers to see the catalog of items, and select items and declare a date range of desired use.”

Eric - “OK, that's most of the work done, and we can use standard tools or even re-use something from the past. Then we will need to publish these “Need” Datastream records.”

Ann - “Yes, and they will be serviced just like that #129 claw hammer that Norm rented. Except the date is not today, it is a future date.”

CHAPTER TWENTY-SEVEN

In Which

ACME RENTALS Gains an Inter-Store Transport System

Norm - “Now we will need a method for items to be transported between stores to meet the reservation commitments, and to balance stock in general. This is key to Acme's operating costs. If sharing is not efficient then Acme will have to buy too much inventory.”

Eric - “Yes, this will be a key aspect of our sales pitch to Acme management.”

Ann - “OK, so the reservation system will publish a Datastream record for each item transport – stating the Item #, the source store and the destination store. A second engine might aggregate these transport records and create a manifest for the shipper/receiver staff.”

Norm - “This is a tricky area. I expect we will create 2 or more methods using different engines and test them for best performance.”

Eric - “I like that! Because the engines are cheap to write, we can be playful with them and try different approaches and philosophies. That is HUGE!”

CHAPTER TWENTY-EIGHT

In Which

The Team Tackles Asset Management

Norm - "The Department of National Defence has asked our assistance with their asset management project. It began in 2001 and is now scheduled to complete in late 2013. The DND is seeking a new direction and they think we may be able to help."

Eric - "Yes, I read that they have spent many tens of millions on this and have continually failed. What has gone wrong, I wonder? How is this seemingly mundane goal so elusive?"

Ann - "I suspect they have always approached it as a database problem. But they are always facing anarchy in their space. I mean, their assets are always on the move and getting blown up and such."

Phil - "I'm certain we all agree that asset management is a perfect assignment for Datastream. It's a walk in the park for our Datastream paradigm."

Norm - "Eric, can you suggest a project plan for this, off the top? Do you need a 10 minute break to collect your thoughts?"

Eric - "Surely you jest! Stay seated - I do not need 10 minutes to analyze this application!

Step 1 is to tag every asset at receiving, before payment is authorized to the

Supplier. The tag can be RFID, or optical Barcode, etc.

Step 2 is to provide lots of RFID / Barcode readers that can report to the Cloud whenever they detect an asset tag. They will append the GIS info and the time and emit a Datastream record just to say, “This asset is here, now.”

Step 3 is to collect all these Datastream records in the Cloud and store them indelibly in a Datastream repository, like we always use.

Step 4 is to write Engines that crawl over the Datastream repository to create reports and perform analysis.”

Norm - “Sounds about right. What are the chances that we would fail at this project, like all those who have gone before us?”

Ann - “You're joking, right? How could we mess this up? It's so simple and direct.”

Phil - “Our 'silver bullet' is that the Datastream paradigm does not need a master plan in order to succeed. We do not need to interview every stakeholder in the military in order to get started . We just create the infrastructure to capture and record the presence of an asset at a location at a time and then send that Datastream record into the Cloud to be indelibly recorded. Nothing more is required. All the analysis will be deferred to engines which can be written for any purpose anytime in the future.”

Norm - “How is it possible that we can have full agreement on a plan after 10 minutes of discussion, when all the teams that have gone before us have failed?”

Eric - “It's the power of the Datastream paradigm, Boss. It just eats problems for breakfast.”

CHAPTER TWENTY-NINE

In Which

The Team Tackles a Gun Registry

Norm - "The Quebec government has asked our assistance with their gun registry project. The federal project began in 1995 and has absorbed close to $1 billion. Quebec would like to spend $1 million to build a new provincial registry. They think we may be able to help."

Phil - "I'm certain we all agree that a gun registry is a perfect assignment for Datastream. It's a walk in the park for our Datastream paradigm."

Norm - "Eric, can you suggest a project plan for this, off the top? Do you need a 10 minute break on this one to collect your thoughts?"

Eric – "Surely you jest! Stay seated - I do not need 10 minutes to analyze this application either!

Step 1 is to record the gun make and model number, serial number, and the Social Insurance Number and Drivers Licence number of the gun owner. This can be done at the registration office – probably a police station or hunting licence office, etc. Print a receipt and tell him to come back tomorrow for his registration.

Step 2 is to form a Datastream record by appending the GIS info and the time and publish into the Cloud a Datastream record saying, "This gun owner is seeking to register this gun here, now."

Step 3 is to subscribe to approve/deny Datastream records from any engines that are empowered to do that.

Step 4 is for the issuing office to print the registration documents if there are no denials after all the votes are in.

Step 5 is to write engines that crawl over the Datastream repository to create reports and perform analysis."

Norm - "Sounds about right. What are the chances that we would face major cost overruns in doing this project, like those who have gone before us?"

Eric - "Well, a late arriving change might involve the court system being able to deny a gun registration if the gun model is prohibited, say. Or the owner has unpaid fines."

Phil - "Not a problem. We'd just create an engine to subscribe to the registration Datastream record types, and then perform its search logic, and publish an approve or deny record. The issuing office engine would be programmed to subscribe to these records and proceed accordingly. Every conceivable change along these lines could be easily accommodated."

Ann - "It's so simple and direct."

Phil - "Our 'silver bullet' is that the Datastream paradigm does not need a master plan in order to succeed. We do not need to interview every stakeholder in the Quebec police and court systems in order to get started. We just create the infrastructure to capture and record the request to register a gun at a location at a time and then send that Datastream record into the Cloud to be indelibly recorded. Nothing more is required. All the analysis will be deferred to engines which can be written for any purpose any time in the future."

Norm - “How is it possible that we can confidently endorse a plan after 10 minutes of discussion, without interviewing the Quebec stakeholders?”

Eric - “It's the power of the Datastream paradigm, Boss. It just eats problems for breakfast.”

CHAPTER THIRTY

In Which

The Team Concludes That Datastream is Agile

Norm - "We have an RFQ from the City. We can do the work easily, and we will certainly be the low bidder, because we can develop a compliant system in the fewest person-hours than any competitor. BUT the City insists that they will only consider vendors who are using Agile Programming methods. "

Ann - "Hmmm, are we?"

Eric - "I reviewed their 'Twelve Principles of Agile Software' document and I believe we are covering every item they have listed, and more.

1. Satisfy the customer through early and continuous delivery
2. Welcome changing requirements, even late in development
3. Deliver working software frequently
4. Business people and developers work together daily
5. Build projects around motivated individuals
6. Convey information via face-to-face conversation
7. Working software is the primary measure of progress-
8. Maintain a constant pace indefinitely
9. Give continuous attention to technical excellence

10. Simplify: maximize the amount of work not done

11. Teams self-organize

12. Teams retrospect and tune their behaviors”

http://agileinaflash.blogspot.ca/2009/08/12-principles-for-agile-software.html

Phil - “I agree. We are especially good at this requirement '*Simplicity--the art of maximizing the amount of work not done--is essential*.' “

Norm - “OK, I will write up our quote and submit it tomorrow.”

CHAPTER THIRTY-ONE

In Which

The Team Tames a Lion

Norm - “Here's an interesting new request. The Lions Club is trying to archive its historical photos, scanned documents, videos and such. They need to secure this material against damage, and it's important to identify the Lions in old photos because the knowledge is leaving their clubs as people pass on.”

Ann - “This is a good deed, Norm. And every service club has the same problem.”

Eric - “I have seen boxes and boxes of material come from estates. The clubs have no place to store this material. They must get it into digital format or it will be lost!”

Phil - “I know they have a problem with choosing image formats, and they need some type of online storage so they aren't dependent on a drive that is passed from member to member.”

Norm - “OK, let's make a Datastream plan to save the Lions' history. Eric, it's your turn to take the lead.”

Eric - “Right, boss. Well we know the drill by now.

Step 1 - Create Datastream records to describe each archive file.

Step 2 - Establish an email address where members can send scanned image

files as attachments. They can use Dropbox to copy the files out to the Cloud storage. This will avoid the slow upload speeds. Each file will have a unique number in the file name and it will always be accessible by that identifier.

Step 3 - The librarian volunteers will capture all data they can about each file item, and record this data in the Datastream records.

Step 4 - Build a few search engines to assist with searching the archive for materials of interest. Most often the search will be for a member's name. But also for documents like rosters, budget documents, advertising flyers, event agendas, sales tax filings, Reports to the Lions head office, etc."

Ann - "Eric, where will the unique file numbers come from?"

Eric - "There could be an engine that issues these, assuring that 2 files never have the same ID. Perhaps the Dropbox service can assign the ID as each new file arrives in the Cloud? At any rate, we can't burden the volunteers with this task - it's too important and needs to be done centrally."

CHAPTER THIRTY-TWO

In Which

The Team Controls Traffic Signals

Norm - “Well, here's an RFQ that we will have to pass on. It's for a traffic control computer for the City. That's a very specialized technology and we have no clue how to approach it, right team? “

Eric – “Whoa, not so fast there Chief! This is a pet project of mine – I think about to all the time while I'm out driving.”

Ann - “OK, Boy Wonder, just how are you gonna stretch Datastream to fit this gig?”

Eric - “As always, Datastream is so simple.

Step 1 – Realize that what drivers want is to join a cohort at a red light, and have that cohort see only green lights for its whole journey on a major street. This minimizes idling at reds and reduces pollution.

Step 2 – The signal controllers that the City has installed are “synchronized” in groups of up to 10 Intersections. The problem is that the groups are not synchronized, and the controllers may be set to hasten the green for a car on a side street. This is bad because it breaks the flow of the cohort, and often creates a new cohort by cutting the original cohort in two.

Step 3 – Publish Datastream records when each signal turns red, green, and when a pedestrian call button is pressed.

Step 4 – Create logic engines to monitor the experience of each cohort on all major streets. These engines can publish Datastream records to request that a RED phase be lengthened by a few seconds, in order to re-synch a cohort. This is a gentle nudge in the right direction.

Step 5 – Create a supervisor logic engine to analyze all these nudges and publish Datastream records to instruct the controllers to alter their timing.

That's about it. The historical Datastream records can be analyzed after the fact to assess the overall benefits, and to detect opportunities for improvement in the engine algorithms. This will be different for each city."

Norm - "OK, I get it now. Good work, Eric. You are proposing a supervisory system that gently nudges the phase of the lights into the best synchronization for the traffic flow."

Phil - "Wow, you could alter the algorithms for time of day, to facilitate rush hour periods."

Ann - "And at night you could shorten the RED periods since the cohort size is smaller and clears a light in less time."

Eric - "What about car present loops? Can we use Datastream records from these?"

Phil - "Sure, does no harm to have extra information. I'm sure that some of the engines will find a way to use those traffic clues."

CHAPTER THIRTY-THREE

In Which

The Team Wrestles with BPM, DCM and CRM

Ann - "Norm, I was reading this IBM-sponsored article last night, about strategies for building the best case-management architecture: http://docs.media.bitpipe.com/io_10x/io_101940/item_478548/IBM_ebizQ_LI%23478548_E-Guide_111111.pdf

Frankly, it seems that we might be missing the boat on BRP (Business Process Management) and DCM (Dynamic Case Management) and CRM (Customer Relationship Management). Do they know something we don't?"

Norm - "Funny, I read the same paper last night! And I had a hard time comparing our paradigm to theirs. It seemed like they are all wound up in unproductive debate, while we are striding confidently ahead."

Brian - "Au contraire, I think we know something they don't. The proof of the pudding is in the eating. Over the past 3 months our team using our Datastream paradigm has produced many more robust software solutions than they could imagine in their wildest dreams. If they had observers in our camp, they would report back, incredulous. ***How do those Datastream dudes get so much done? How do they manage to sidestep the process morass that bogs us down at every turn?"***

Eric - "So, what is our BPM? I guess it would be: to indelibly record every event that might be of interest, and then analyze these records by means of independent

logic engines, which can employ hidden databases.“

Norm - “Yes, that's well said, Eric. That is how we have approached every project, and it has worked well for us. It seems that the gulf separating our paradigms is so vast that we live in separate worlds. Truly – I think they are dinosaurs. Developer teams that use our paradigm will typically under-quote teams that use their paradigm by at least an order of magnitude!”

CHAPTER THIRTY-FOUR

In Which

The Team Addresses Database Atomicity

Phil - "Norm, I think we may have a big problem! How are we going to handle the requirement to ensure that a group of reserved seats is removed from seat inventory only when payment is complete? How are we going to reverse the seat removal from available seat Inventory, if payment fails?"

Norm - "Well, Phil, I see why the database guys have troubles with this, but why do you fear that we can't handle it? All we ever do in Datastream is record the sequence of events. If the last step in the reservation process - payment - fails to complete then this is apparent with the passage of time (in which case the transaction aborts), or with an explicit transaction failed to complete Datastream message."

Phil - "So you're saying that it's up to the seat inventory module to watch for payment to complete?"

Norm - "Yes. When anybody wants to audit for Atomicity, they would naturally look at the seat inventory module to see how it was implemented. If the logic is not there or not properly handled, then we know exactly where to make the logic improvements. This is better than leaving it to the database utilities where we will never see how it is done."

Phil - "OK, I see. And in our Datastream world we can always examine historical Atomicity if there's a problem reported, because Datastream is a perpetual log of

time-stamped events."

Norm - "Yes. And we can test various candidate modules to see which performs best in a given environment. We can optimize a module to an extreme extent if we choose to. For example, imagine a module that down-counts a finite resource like reserved seats, and uses a C array in memory. This would run amazingly fast, because it is purpose-built and has no other constraints upon it."

Phil - "What I like about the Datastream approach is that every processing module exists in its own bubble. So the modules that form a project can employ many different technologies over the years. And the technologies used can be in constant flux. Now it seems to me that Datastream is object oriented computing, using a paradigm that we can live with and prosper with!!"

CHAPTER THIRTY-FIVE

In Which

The Team Addresses Race Conditions

Phil - "Norm, how are we going to handle race conditions - you know, like selling stocks on an exchange, say?"

Norm - "OK, let's consider a finite resource like a new stock issue, an IPO. As the sales get down to the last few shares, how shall we avoid over-selling these shares."

Eric - "We would build a module that would grant each share purchase and reduce the finite number of shares available for sale. Requests would be handled on a first-come basis. Each request is time-stamped. If multiple requests have the same time down to the millisecond, then the first of this Datastream record that the module processes will get the last shares. And maybe not get all the shares that were requested."

Norm - "OK. And you could treat the race like a photo-finish at the horse track if it was crucial. A judge can do this by reviewing the Datastream traffic at the end."

CHAPTER THIRTY-SIX

In Which

The Team Tackled Insurance

Norm - “We just got a sales enquiry from the Insurance industry. What do you fellows think?”

Eric – “I think we can make a very good system for insurance, in Datastream.”

Brian – “That's goofy. It takes years to do an insurance system. I know, I've been there”

Eric – “Granted. But now we're a Skunkworks, and we perform miracles every day. Besides, insurance is basically simple. Fill in a policy application, rate the risk, and collect the premium. When claims come in, you fill in a claim application, check that the policy is paid up and covers the loss, then cut a cheque. Insurance is not rocket science.”

Brian – “I can't really argue with that, Eric. Insurance is basically simple - but the devil is in the details, as always!”

Norm - “Well, let's give it two weeks and see if it yields easily or not.”

The team made rapid headway in establishing the first CD record formats, for life insurance. Client ID, Policy #, beneficiary, amount, medical declarations, etc. All items that are “of interest” and printed on a policy. Then they added fields for premium payment history and some underwriting data.

CHAPTER THIRTY-SEVEN

In Which

The Team Built a Supermarket

Product Locator Terminal

Norm - “We just got a sales enquiry for a customer-operated terminal for use in a Supermarket, to assist customers in finding food items in the store.”

Brian - “Now that's a great idea! I would use that every week, at least. And I would prefer the store that had this system!”

Norm - “OK, let's get started. I can see a simple touchscreen GUI shoebox for the customer to operate. He would start spelling the item he wants, and another shoebox with access to a product database would generate several proposals until the customer selects one of them.”

Eric - “OK, the item database could come from the store's POS system, so it is gifted to us. We could take an extract once a week, say. Or once a day, whenever the shelf locations change.”

Phil - “So we need CD/DS records to deliver the letters as they are entered by the customer.”

Norm - “Yes, for example A - R - T - I - C - H - O - K - E - H - E - A - R - T - S. But generally the first 2 or 3 letters will bring the desired item into view.”

Brian - “When the customer spies her item, she selects it. And then the aisle and location appears. She can print a slip to take with.”

Norm - “The user might have a list of 10 items in a session. The aisles would be clearly numbered. A store map printout would be nice. Perhaps mark a route from the

terminal to each item?”

Ann - “You know this customer terminal, why not make that a Smartphone?”

Brian - “Great concept! Let's do that.”

Norm - “OK, we'll build a bare-bones proof of concept and we can embellish it later when we have a contracted client.”

That night Norm pondered and reflected on his Skunkworks project, his Datastream paradigm and all that he had witnessed over the past few weeks. His team had embraced the paradigm, slowly at first and with a healthy degree of skepticism. To their credit, they gave it a chance and wound up committed to it wholeheartedly.

Over time, the team had begun to suggest and create uses for this new software development system. This confirmed what Norm had suspected in that the paradigm was transferable and easily mastered if people allowed themselves to accept something new and revolutionary and then get involved. Development going forward did not depend entirely on his own creativity. For it to truly take hold and become the new standard, this was vital.

Norman sighed again. This time it was a sigh of relief and satisfaction.